Inteligencia Artificial

Álvaro Fernández García

PRÁCTICA 1: AGENTES REACTIVOS: MEMORIA

En esta memoria se pretende explicar detalladamente el comportamiento que se ha definido para el agente de la Práctica 1. Por tanto el documento se divide en las siguientes partes:

1. Resumen general del comportamiento.
2. Variables privadas del agente.
3. Método think.
4. Funciones extra.

1.- RESUMEN GENERAL DEL COMPORTAMIENTO

Se trata de un agente reactivo con memoria. El comportamiento que sigue para recorrer los mundos de Belkan se basa en avanzar mientras pueda, es decir mientras no se encuentre con un obstáculo que le impida el paso. Si se produce este caso, el agente girará aleatoriamente en una dirección, (izquierda o derecha). También se producen giros cuando el agente ha realizado un número determinado de avances. Aunque no tenga ningún obstáculo delante, si ese número de avances se ha alcanzado, el agente girará en una dirección aleatoria.

Por otra parte el agente podrá ir recogiendo objetos del mapa para posteriormente guardarlos en la mochila hasta que sea necesario su uso. Podrá recoger un máximo de 4 objetos, cada uno de un tipo. Si se encuentra con un tipo de objeto que ya tiene en su mochila, el agente lo tomará como un obstáculo y lo esquivará. Cuando necesite un objeto, sacará de la mochila uno, si no es el que busca lo volverá a guardar y sacará otro. Anteriormente se había realizado otra alternativa en la que el agente, cuando buscaba un objeto, si extraía de la mochila un objeto que no era el que buscaba lo arrojaba, perdiendo dicho objeto. Sin embargo este comportamiento presentaba un problema. Cuando por ejemplo nos encontramos en el bosque y delante tenemos un lobo, el agente comenzará a buscar un hueso en su mochila, y para ello deberá tirar las zapatillas, por tanto se quedará atrapado en el bosque sin poder continuar. Por eso se ha optado por la otra solución en la que no se pierden objetos.

Las situaciones en las que el agente buscará un objeto se detallarán más adelante. El único objeto que pierde es el hueso si se lo da a un lobo. Los demás se mantienen hasta que se reinicie.

Por último mencionar que el agente tiene prioridad por buscar las casillas PK (si todavía no ha encontrado una anteriormente), para ello podrá fijarse objetivos y calcular una ruta hasta ellos.

2.- VARIABLES PRIVADAS DEL AGENTE

Las variables privadas que presenta el agente y su descripción son las siguientes:

- Fila, columna y brújula → el agente las utiliza para conocer en todo momento la posición y la orientación que presenta dentro del mapa. En su inicio tendrán el valor por defecto de 99, 99 y orientación hacia el norte. Su valor se actualiza con cada paso, y cuando se alcanza un punto PK, fila y columna toman los valores adecuados en el mapa.
- Girar_en → esta variable contiene el número máximo de avances que puede realizar el agente. Una vez que se ha alcanzado este número, el agente realizará un giro aleatorio en una dirección, reiniciando también la cuenta. Su valor se ha determinado empíricamente, seleccionando aquel con el que se han obtenido mejores resultados. El valor es 61.
- ultimaAccion → como se trata de un agente con memoria, este almacena la última acción que ha realizado y la tiene en cuenta para la toma de decisiones.
- girarDerecha → es un booleano cuyo valor se calcula aleatoriamente, si está activo a la hora de tomar una decisión de giro, girará a la derecha, si no esta activo girará a la izquierda
- bien_situado → booleano que se activa cuando el jugador ha pasado por una casilla PK y el valor de fila y columna es correcto.
- memoriaTemp → Se trata de un mapa auxiliar. El agente almacena temporalmente el mapa descubierto mientras no se ha situado en una casilla PK. Tiene un tamaño 200x200, (el tamaño de mapa máximo es de 100 por eso se ha seleccionado el doble del tamaño) También esto explica el valor de fila y columna a 99, al principio está situado en el centro de esta matriz. Cuando pase por un punto PK, el contenido de esta matriz pasará al mapaResultado.
- vfil y vcol → almacenan temporalmente el valor de fila y columna de memoriaTemp, justo antes de traspasarla a mapaResultado.
- objetoNecesario → almacena el objeto que necesita buscar en la mochila el agente. Bikini cuando tenga agua delante, zapatillas si tiene bosque, hueso si tiene un lobo y llaves si tiene una puerta.
- objetos → Se trata de un vector de booleanos, cada posición se corresponde con un objeto. Una posición estará a true si el agente ha recogido el objeto correspondiente, false en cualquier otro caso. Sirve para consultar que objetos ha recogido ya el agente.
- objetivo_fijado → Es un booleano que indica al agente si está siguiendo una ruta fijada. Mientras se sigue una ruta el comportamiento aleatorio no se lleva a cabo.
- pos_objetivo → Almacena la posición del vector sensores.terreno en la que se encuentra el objetivo. Se utiliza para calcular la ruta hacia él.
- Giro y num_avances → el número de avances y el giro (izquierda o derecha) que tiene que realizar para alcanzar el objetivo.
- objetoDado → Cuando el agente entrega el hueso, objetoDado pasa a tener el índice del objeto correspondiente. El objetivo de esta variable es para posteriormente comprobar si efectivamente el objeto ha sido entregado o no para actualizar sin errores el vector de objetos. Puede ocurrir que cuando le des un hueso a un lobo, este se desplace y no lo entregues. Sería un error colocar que no tienes hueso, porque en realidad no lo has perdido..

3.- MÉTODO THINK

El método think presenta la siguiente estructura:

1. Realizar cout de las distintas variables necesarias para controlar el estado de la simulación. A continuación se comprueba si se ha reiniciado el juego. Si es así llama a la función restart.
2. Actualizar la posición del agente en el mapa. Para ello en función de la última acción se modifican los valores de fila, columna y brújula. Es muy importante a la hora de avanzar tener en cuenta que no se ha colisionado con sensores.colision, ya que realmente no nos impide realizar la acción de avance pero el agente no se mueve. Si actualizamos el valor de la posición sin haber avanzado puede dar lugar a errores.
   A continuación se presenta el pseudocódigo.

```
Si avancé una casilla y no he colisionado contra un objeto
      Actualizo el valor de fila y columna en función de mi orientación (N,S,E,O)
Si giré
     Actualizar el valor de la brújula en función de la dirección en la que giré y el valor
original de la brújula
```

3. Añadir el cono de visión al mapa correspondiente. Para ello se utiliza la función aniadeAlMap, la cual se detallará más adelante. A continuación se comprueba si el agente se encuentra sobre una casilla PK y si no estaba bien situado. Si esto se cumple se pasa a realizar el paso de la matriz memoriaTemp a mapaResultado con la función MergeMap y se actualiza el valor de fila y columna. Por último la variable bien_situado se coloca a true para indicar que el agente ya está bien posicionado.
4. A continuación se lleva a cabo la toma de decisiones en función de lo percibido por los sensores. Decir que el orden en el que colocan las sentencias else if le confieren una prioridad a las mismas. Las cláusulas en pseudocódigo son las siguientes:

```
Si no estoy bien situado
     Comprobar si en sensores.terreno hay algún punto PK
     Si lo hay y no está alineado con el agente.
          Calculo una ruta con la función FijaRuta.
Si lo último que hice fue recoger un objeto.
    Lo guardo y actualizo el contenedor “mochila”
Si puedo avanzar por el terreno *
    Si tengo fijada una ruta, la sigo. Realizo el número de avances necesarios y por
    último giro.
    Si no: si no se ha cumplido el máximo de avances, avanzo y actualizo
    el número de avances que me quedan. Si sí se ha cumplido, giro en una dirección
    aleatoria
Si delante tengo un objeto
    Si no he recogido antes el objeto
          Si no tengo nada en las manos lo recojo, si tengo algo guardo lo que tengo.
    Si por el contrario ya tengo ese objeto, giro en una dirección aleatoria
Si lo ultimo que hice fue sacar un objeto de la mochila
    Si no es lo que buscaba vuelvo a guardarlo. En cualquier otro caso no hago nada.
Si delante tengo agua, ningún obstáculo y tengo el bikini puesto.
```

Aplicar los criterios del apartado “puedo avanzar por el terreno”
Si delante tengo agua, tengo el bikini en la mochila pero no lo tengo puesto.
Si tengo las manos vacías saco un objeto de la mochila. Si no guardo el que tenía.
Pongo que estoy buscando el bikini. Este apartado se enlaza con “ultimaAccion = sacar de la mochila” para buscar el objeto.
Si delante tengo bosque, ningún obstáculo y tengo las zapatillas puestas
Aplicar los criterios del apartado “puedo avanzar por el terreno”
Si delante tengo bosque, tengo las zapatillas en la mochila pero no las tengo puestas.
Si tengo las manos vacías saco un objeto de la mochila. Si no guardo el que tenía.
Pongo que estoy buscando las zapatillas. Este apartado se enlaza con “ultimaAccion = sacar de la mochila” para buscar el objeto.
Si delante tengo una puerta cerrada y tengo las llaves en la mano.
Abro la puerta (dar las llaves)
Si delante tengo una puerta cerrada y tengo las llaves en la mochila
Si tengo las manos vacías saco un objeto de la mochila. Si no guardo el que tenía.
Pongo que estoy buscando las llaves. Este apartado se enlaza con “ultimaAccion = sacar de la mochila” para buscar el objeto.
Si delante tengo un lobo y el hueso en las manos
Le doy el hueso.
Indico que he dado el hueso.
Si delante tengo un lobo y tengo un hueso en la mochila
Si tengo las manos vacías saco un objeto de la mochila. Si no guardo el que tenía.
Pongo que estoy buscando el hueso. Este apartado se enlaza con “ultimaAccion = sacar de la mochila” para buscar el objeto.
En cualquier otro caso
Giro aleatoriamente a izquierda o derecha

Si lo último que hice fue dar un objeto
Si tengo las manos vacías lo entregué con éxito, luego actualizo la entrada del vector de objetos

* Entender que se puede avanzar por el terreno cuando delante se tiene suelo arenoso, suelo pedregoso, un punto PK, una puerta y ningún obstáculo delante. (Una casilla de tipo puerta sin obstáculo delante es una puerta abierta).

5. Se actualiza el valor de ultimaAccion y se devuelve la acción elegida.

4.- FUNCIONES EXTRA

Se han utilizado las siguientes funciones para el desarrollo del agente:

- restart → se utiliza para reiniciar el estado del agente, filas y columnas a 99, brújula a N, poner que no se ha recogido ningún objeto, bien_situado a false… Se llama a esta función cuando el agente muere y también en el constructor del mismo.
- MergeMap → pasa los resultados guardados en MemoriaTemp a mapaResultado cuando se llega a un PK. Para ello utiliza fila y columa (que tendrá los valores dados por el PK) y vfil y vcol que contiene la posición en la que se encontraba el agente en MemoriaTemp.
- AniadeAlMap → Función que guarda en el mapa correspondiente (si esta bien situado en mapaResultado y si no en MemoriaTemp) el cono de visión del agente

(vector sensores.terreno). Para ello utiliza la posición del agente (fila, columna y brújula).

- FijaRuta → Cuando el agente detecta un punto PK cerca, no esta bien situado y el PK no está alineado con el debe de calcular una ruta hacia él. Para ello utiliza la posición del vector sensores.terreno en la que está el PK. A partir de ella calcula el número de veces que tiene que avanzar y hacia donde tiene que girar para alcanzarlo.